CLAIMS

What is claimed is:

1. A method comprising:

scheduling an instruction for execution;

speculatively executing said instruction;

determining whether said instruction executed correctly;

routing said instruction to a replay mechanism if said instruction did not execute correctly;

determining whether a replay tornado exists;

routing said instruction for re-execution if said instruction executed incorrectly and no replay tornado exists; and

breaking said replay tornado if said replay tornado exists, said breaking comprising:

retiring replay safe instructions in said pipeline;

marking non-replay safe instructions in said pipeline for re-execution;

and

rescheduling said non-replay safe instructions for re-execution.

2. The method of claim 1 further comprising tracking length of time that has elapsed since a last primary replay cause was detected.

3. The method of claim 2 wherein said determining whether a replay tornado exists further comprises checking if said length of time that has elapsed has exceeded a tornado detection time limit when a secondary replaying instruction is detected.

4. A method comprising:

scheduling an instruction for execution;

speculatively executing said instruction;

determining whether said instruction executed correctly;

routing said instruction to a replay mechanism if said instruction did not execute correctly; and

choosing a replay mechanism that will not put said instruction back into execution

immediately said instruction cannot be executed successfully for a relatively long time.

5. The method of claim 4 wherein an incorrectly executed instruction is not put back into execution until when a signal indicates that said incorrectly executed instruction can be executed correctly.

6. The method of claim 5 further comprising placing said incorrectly executed instruction back into execution in a manner that will not start a tornado.

7. The method of claim 6 wherein said placing said incorrectly executed instruction back into execution in a manner that will not start a tornado comprises:

entering instructions that are not replay safe into a scheduler in program order;

setting a result register for each of said instructions entering said scheduler; and

scheduling said instructions for execution.

8. A method comprising:

storing an instruction in a queue, said queue to store a plurality of instructions and information related to each of said instructions, said information comprising a set of status bits for each of said instructions;

scheduling said instruction;

speculatively executing said instruction;

checking whether said instruction executed correctly;

flagging said instruction for replay if said instruction did not execute correctly;

determining whether a replay tornado exists and if said instruction is part of said replay tornado;

retiring replay safe instructions and

rescheduling said non-replay safe instructions for re-execution.

9. The method of claim 8 further comprising toggling status bits in said queue for said non-replay safe instructions to indicate need to reschedule and re-execute said non-reply safe instructions.

10. The method of claim 9 further comprising tracking amount of time since a last primary replay cause.

11. The method of claim 10 wherein said determining whether said replay tornado exists further comprises comparing said time since said last primary replay cause with a time limit, wherein said replay tornado exists if said time since said last primary replay cause exceeds said time limit when a secondary replaying instruction is detected.

12. The method of claim 11 wherein said determining whether said replay tornado exists further comprises measuring a current rate of replay and determining whether a tornado exists if said current rate of replay exceeds a time limit since said last primary replay cause.

13. The method of claim 9 wherein a primary cause instruction is added to a list of replay tornado causing instructions if said replay tornado exists and said primary cause was said last primary replay cause before detection of the tornado.

14. The method of claim 13 wherein said list of replay tornado causing instructions is used to predict instructions that are likely to cause replay tornadoes.

15. The method of claim 14 wherein instructions determined to have not produced a correct result are put back into execution in a manner that will not start tornadoes if said instructions are predicted to be likely to cause replay tornadoes.

16. The method of claim 15 further comprising:
entering said instructions that are not replay safe into a scheduler in program order;
providing results to said instructions entering said scheduler; and
scheduling said instructions for execution.

17. A processor comprising:
a scheduler to dispatch instructions, said scheduler comprising a scheduler queue to store said instructions and a status associated to each of said instructions, said scheduler to store

each instruction until that instruction is indicated to be replay safe;

an execution unit coupled to said scheduler, said execution unit to execute said instructions;

a checker coupled to said execution unit, said checker to determine whether each instruction has executed correctly; and

a replay mechanism coupled to said checker, said replay mechanism to receive an indication from said checker for each instruction that has not executed correctly, said replay mechanism further comprising logic to determine whether a replay tornado exists and whether said incorrectly executed instruction is part of said replay tornado, said replay mechanism further comprising a tornado stopping mechanism to break said replay tornado.

18. The processor of claim 17 wherein said tornado stopping mechanism comprises logic to retire replay safe instructions in an execution pipeline, and to set statuses for said non-replay safe instructions to indicated a need for rescheduling said non-replay safe instructions.

19. The processor of claim 18 wherein said status associated to each of said instructions comprises a set of bits to indicate if associated instruction is replay safe, is part of a replay tornado, is being replayed, and has been dispatched.

20. The processor of claim 19 wherein said indication from said checker causes said replay mechanism to request said scheduler to reschedule said incorrectly executed instruction for re-execution if no replay tornado is present.

21. The processor of claim 20 further comprising:

a retirement unit coupled to said checker to receive any instructions that have executed correctly, said retirement unit to retire said instructions that have executed correctly;

a first level internal cache coupled to said execution unit; and

a second level internal cache coupled to said execution, wherein access time to said second level internal cache is greater than to said first level cache.

22. The processor of claim 21 wherein said processor is coupled to an external main memory

and to a disk memory.

23. The processor of claim 22 wherein said incorrectly executed instruction is a memory load operation, said memory load instruction operation to cause a memory fetch, said memory fetch to search through a memory hierarchy for requested data, wherein said memory hierarchy comprises of said first level cache, said second level cache, said external main memory, and wherein said first level cache has fastest access time and said main memory has longest access time; and wherein each incorrect execution of said memory load instruction causes said memory fetch to access a slower level of memory, and said logic increases said time delay to approximate an access latency to said slower level of memory.

24. A system comprising:

a memory coupled to a bus;

a processor coupled to said bus, said processor comprising:

a scheduler to dispatch instructions, said scheduler comprising a scheduler queue to store said instructions and a status associated to each of said instructions, said scheduler to store each instruction until that instruction is indicated to be replay safe;

an execution unit coupled to said scheduler, said execution unit to execute said instructions;

a checker coupled to said execution unit, said checker to determine whether each instruction has executed correctly; and

a replay mechanism coupled to said checker, said replay mechanism to receive an indication from said checker for each instruction that has not executed correctly, said replay mechanism further comprising logic to determine whether a replay tornado exists and whether said incorrectly executed instruction is part of said replay tornado, said replay mechanism further comprising a tornado stopping mechanism to break said replay tornado.

25. The system of claim 24 wherein said tornado stopping mechanism comprises logic to retire replay safe instructions in an execution pipeline, and to set statuses for said non-replay safe instructions to indicated a need for rescheduling said non-replay safe instructions.

26. The system of claim 25 wherein said replay mechanism further comprises a timer to track a time since a last primary replay cause instructions, wherein said tornado stopping mechanism uses said time since said last primary cause when a secondary replaying instruction is executing to determine whether said replay tornado exists.

27. The system of claim 26 wherein said indication from said checker causes said replay mechanism to request said scheduler to reschedule said incorrectly executed instruction for re-execution in a tornado free manner.